מטלת מנחה 14 - מבני נתונים ומבוא לאלגוריתמים

שאלה 1

סעיף א

ראשית, נחשב כי מספר הקלטים האפשריים למערך בגודל 4 הוא $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$.
נחשב, לפי מודל עץ ההחלטות, חסם תחתון למספר ההשוואות. במודל זה, מספר ההשוואות המינימלי הנדרש כדי למיין מערך חסום מלמטה על ידי גובה העץ, שנסמנו ב-h.

מאחר ולעץ יש 24 עלים, ובכל קומה מספר העלים חסום מלמעלה על ידי 2^i, בקומה ה-h מספר העלים יהיה חסום מלמעלה על ידי 2^h.

נפעיל את הפונקציה הלוגריתמית, שהיא מונוטונית עולה, על אי השוויון $2^h \geq 24$ ונקבל את אי-השוויון $h \geq \lg 24 \approx 4.585$. מאחר ו-$h$ מספר שלם (לא ייתכן שתהיה לעץ חצי קומה), נקבל $h \geq 5$ ובהתאם מספר ההשוואות הנדרשות כדי למיין מערך בגודל 4 חסום מלמטה על ידי 5, כנדרש.

נציע אלגוריתם הממיין מערך בגודל 4 ב-5 השוואות. נניח כי האינדקס הראשון במערך הוא 1.

$Sort\_Four\_Elements(a)$
 $\triangleright Ensure\ a[1] \leq a[2]$
 $if\ not\ a[1] \leq a[2]$
  $swap(a, 1, 2)$
 $\triangleright Ensure\ a[3] \leq a[4]$
 $if\ not\ a[3] \leq a[4]$
  $swap(a, 3, 4)$
 $\triangleright Ensure\ a[1] \leq a[3]$
 $if\ not\ a[1] \leq a[3]$
  $swap(a, 1, 3)$
 $\triangleright Ensure\ a[2] \leq a[4]$
 $if\ not\ a[2] \leq a[4]$
  $swap(a, 2, 4)$
 $\triangleright Ensure\ a[2] \leq a[3]$
 $if\ not\ a[2] \leq a[3]$
  $swap(a, 2, 3)$

ניתוח נכונות: יהא קלט $[a, b, c, d]$.
לאחר ההשוואה הראשונה: $[min(a, b), max(a, b), c, d]$
לאחר ההשוואה השנייה: $[min(a, b), max(a, b), min(c, d), max(c, d)]$

נשים לב: $min(min(a, b), min(c, d)) = min(a, b, c, d)$,
לכן, לאחר ההשוואה השלישית, $[min(a, b, c, d), max(a, b), max(min(a, b), min(c, d)), max(c, d)]$

נשים לב: $max(max(a,b), max(c,d)) = max(a,b,c,d)$
לכן, לאחר ההשוואה הרביעית,
$[min(a,b,c,d), min(max(a,b), max(c,d)), max(min(a,b), min(c,d)), max(a,b,c,d)]$
כעת, אנחנו בטוחים כי האיברים במקום ה-1 וה-4 הגיעו למקומם הנכון בעת המיון. ברור כי כעת מתקיים $a[1] \le a[2], a[3], a[4]$ וכן $a[4] \ge a[1], a[2], a[3]$.

כל מה שנותר הוא לקבוע את הסדר היחסי בין האיברים $a[2], a[3]$ כעת. נעשה זאת ע"י השוואתם והחלפה במקרה הצורך - זו ההשוואה החמישית. כעת מתקיים $a[2] \le a[3]$, ולפי טרנזיטיביות $a[1] \le a[2] \le a[3] \le a[4]$ כנדרש.

סעיף ב

על מנת למצוא חסם תחתון, מספיק לחסום מלמטה את מספר הקלטים האפשריים, ולהסיק לפי מודל עץ ההחלטות חסם תחתון למספר ההשוואת עבור קלטים אלו, ובהתאם לסיבוכיות זמן-הריצה. לאחר מכן, על מנת להוכיח שהחסם הדוק, נמצא אלגוריתם הממיין מערך כזה שסיבוכיות זמן-הריצה שלו היא כמו בחסם התחתון.

מספר הקלטים העונים להגדרה "מערך ממוין עם שגיאה בגודל k" חסום מלמטה על ידי מספר הקלטים העונים להגדרה של שאלה 8.1-4 עבור מערך המחולק ל-$\frac{n}{k}$ תת-סדרות בגודל k כל אחת, כך שכל האיברים בתת-סדרה מסוימת גדולים מכל האיברים בתתי-הסדרות לפניה וגדולים מכל האיברים בתתי-הסדרות אחריה. כלומר:

טענה: כל מערך העונה להגדרה משאלה 8.1-4 הוא מערך ממוין עם שגיאה בגודל k.
הוכחה: יהא מערך בגודל n המחולק ל $\frac{n}{k}$ סדרות בגודל k כל אחת המקיים את תנאי ההשוואה עבור הסדרות כנדרש.

אז יהיו שני איברים a_i, a_j כך ש $j - i > k$. אז ברור כי a_j שייך לתת-סדרה אחרת, מ"ימין" לתת-הסדרה אליה שייך a_i, ולכן לפי תנאי ההשוואה בין שני תתי-הסדרות יתקיים $a_j > a_i$ כנדרש.

כעת, ניעזר בקומבינטוריקה ונמצא את מספר הקלטים האפשריים בגודל n הממלאים את התנאי בשאלה 8.1-4, ובהתאם חסם תחתון למספר הקלטים האפשריים הממויינים עם שגיאה בגודל k. ברור כי מספר זה קטן מ $n!$, שכן לא כל קלט יקיים את התנאי.

תהא סדרה בגודל k. מספר האפשרויות לסידור שלה - $k!$.
כעת, עבור $\frac{n}{k}$ סדרות, מספר הקלטים האפשריים הוא $k! \cdot k! \cdot ... \cdot k!$ - מכפלה של $k!$ בעצמו $\frac{n}{k}$ פעמים. לכן, מספר הקלטים האפשריים הוא $(k!)^{\frac{n}{k}}$.

ניעזר במודל עץ ההחלטות כדי לחסום מלמטה את מספר ההשוואות על קלטים אלו. מספר העלים בעץ הוא $(k!)^{\frac{n}{k}}$, ונסמן את גובה העץ ב-h. מספר ההשוואות חסום מלמטה על ידי גובה העץ בהתאם למודל עץ ההחלטות. כמו כן, מספר העלים בעץ בינארי הוא לכל היותר 2^h, ומכן נסיק את אי-השוויון $2^h \ge (k!)^{\frac{n}{k}}$.

כעת, נפעיל את פונקציית הלוגריתם על אי-השוויון. אי-השוויון נשמר כי פונקציית הלוגריתם היא מונוטונית עולה. לכן, $h \geq \lg(k!)^{\frac{n}{k}} = \frac{n}{k}\lg(k!)$.

נחשב:

$$\lg(k!) = \lg(k(k-1)(k-2)...(2)) =$$
$$= \lg k + \lg(k-1) +....+ \lg 2 = \sum_{i=2}^{k} \lg i =$$
$$\sum_{i=2}^{\frac{k}{2}-1} \lg i + \sum_{i=\frac{k}{2}}^{k} \lg i \geq \sum_{i=2}^{\frac{k}{2}-1} \lg i + \sum_{i=\frac{k}{2}}^{k} \lg \frac{k}{2} =$$
$$\sum_{i=2}^{\frac{k}{2}-1} \lg i + \frac{k}{2}\lg\frac{k}{2} \geq \frac{k}{2}\lg\frac{k}{2}$$

לכן,

$$h \geq \frac{n}{k} \cdot \frac{k}{2}\lg\frac{k}{2} = \frac{n}{2}\lg\frac{k}{2} = \Omega(n \lg k)$$

המסקנה היא שמספר ההשוואות במיון מערך ממוין עם שגיאה בגודל k חסום מלמטה ע"י $\Omega(n \lg k)$, ובהתאם גם סיבוכיות זמן הריצה של אלגוריתם מיון מבוסס-השוואה של מערך כזה חסומה מלמטה על ידי $\Omega(n \lg k)$.

נוכיח כי חסם זה הוא הדוק - נצביע על אלגוריתם למיון מערך ממוין עם שגיאה בגודל k שסיבוכיות זמן-הריצה שלו היא $\Theta(n \lg k)$. קיים אלגוריתם כזה, והוא האלגוריתם המופיע בפתרון שאלה 1 בממ"ן 12. נחזור בקצרה על עיקרי פתרון שאלה זו:

רעיון האלגוריתם:

- נבנה ערמת מקסימום בגודל $k + 1$ מ-$(k + 1)$ האיברים האחרונים בקלט.
- נוציא את האיבר המקסימלי מהערמה ונשים אותו במקום ה-n. ברור כי איבר זה הוא המקסימלי במערך כי לא ייתכן שקיים איבר שהאינדקס שלו קטן מ$n - k - 1$ והוא גדול מהאיבר שהיה במקום ה n (לפי הנתון על המערך). האיבר שהיה במקום ה n, שהיה בערמה בעת הוצאת האיבר המקסימלי, קטן או שווה לאיבר המקסימלי.
- נוסיף את האיבר במקום ה $n - k - 1$ לערימה.
- נחזור על התהליך - נוציא את האיבר המקסימלי בערימה, נשבצו במקום במתאים ונוסיף את האיבר הבא במערך עד אשר לא נותרו לנו איברים להוסיף.
- כעת, נוציא את האיברים שנותרו בערימה ונשבצם במקום המתאים ב-$k + 1$ המקומות הראשונים במערך.

ניתוח סיבוכיות זמן-הריצה של אלגוריתם זה:

ההוראה	סיבוכיות זמן-הריצה
בניית ערימה בגודל $k + 1$	$O(k + 1) = O(k)$
$n - k$ פעמים, בצע הוצאת איבר מהערימה והוספת איבר אחר	$(n - k) \cdot 2 \lg k = O((n - k) \lg k)$
$k + 1$ פעמים, הוצא את האיבר המקסימלי מהערימה	$k \cdot \lg k = O(k \lg k)$

סך סיבוכיות זמן הריצה, כאשר נניח $k \leq n \leq n \lg k$ ולכן $k = O(n \lg k)$:

$$k + (n - k) \lg k + k \lg k = k + n \lg k = O(n \lg k)$$

מאחר ו $\Omega(n \lg k)$ חסם תחתון על סיבוכיות-זמן הריצה של האלגוריתם, נקבל כי חסם עליון זה הוא הדוק, וסיבוכיות זמן-הריצה של אלגוריתם זה היא $\Theta(n \lg k)$ כנדרש.

שאלה 2

סעיף א

נבנה את האלגוריתם שלנו בהשראת אלגוריתם מיון ספירה.
רעיון האלגוריתם:

1. נבנה מערך מונים בגודל k, ונספור כמה מופעים של כל איבר מ-0 עד k יש במערך.
2. נבצע סכום של כל איבר במערך עם האיבר שלפניו, כך שכעת בכל תא במערך המונים תהיה ספירת כל האיברים שערכם קטן או שווה לאינדקס תא זה.
3. בהינתן קלט (a, b), נמצא כמה איברים קטנים או שווים ל-b (ע"י גישה לתא הb במערך המונים), ונחסר את מספר האיברים הקטנים מ-a (ע"י גישה לתא ה $a - 1$ במערך המונים).

עבודת העיבוד

$Make\_Counter\_Array(a,\ k)$

$\quad Counters\ \leftarrow new\ array[k]$

$\quad \triangleright\ Populate\ counter\ array$

$\quad for\ i\ \leftarrow 0\ to\ a.length$

$\quad\quad Counters[a[i]] \leftarrow Counters[a[i]]\ +\ 1$

$\quad \triangleright\ Sum\ the\ counters$

$\quad for\ i\ \leftarrow 1\ to\ k$

$\quad\quad Counters[i] \leftarrow Counters[i]\ +\ Counters[i\ -\ 1]$

$\quad return\ Counters$

מענה על שאילתות

$Elements\_In\_Range(a,\ min, max)$

$\quad if\ min\ =\ 0$

$\quad\quad return\ Counters[max]$

$\quad return\ Counters[max]\ -\ Counters[min\ -\ 1].$

ניתוח נכונות

לאחר הלולאה הראשונה בעיבוד המקדים, ברור כי בכל תא במערך המונים מספר המופעים של האינדקס שלו במערך הקלט.
כעת, נרצה להוכיח את השמורה הבאה. נסמן ב $count_i$ את מספר המופעים של הערך i במערך הקלט: לאחר האיטרציה הi במערך, $Counters[i]\ =\ \sum_{k=0}^{i} count_k$ (הוכחה בהמשך).

אילו השמורה נכונה, אז בסוף העיבוד המקדים יתקיים לכל i, $Counters[i]\ =\ \sum_{k=0}^{i} count_k$.

כעת, בהינתן a ו b כלשהם:

$$Counters[b]\ -\ Counters[a\ -\ 1]\ =\ \sum_{k=0}^{b} count_k - \sum_{k=0}^{a-1} count_k = \sum_{k=a}^{b} count_k$$

וסיימנו.

<u>שמורת לולאה:</u> לאחר האיטרציה הi במערך, $Counters[i] = \sum_{k=0}^{i} count_k$.

נשים לב כי עבור $i = 0$ מתקיים $\sum_{k=0}^{i} count_k = count_0$ כנדרש ולכן הטענה נכונה גם עבור $i = 0$.

<u>בסיס האינדוקציה:</u> עבור $i = 1$,

השגרה תבצע $\sum_{k=0}^{i} count_k = count_1 + count_0 = Counters[1]$ כנדרש.

<u>שלב האינדוקציה:</u> נניח כי עבור $i - 1$, $\sum_{k=0}^{i-1} count_k = Counters[i - 1]$

אז עבור i השגרה תבצע $\sum_{k=0}^{i} count_k = \sum_{k=0}^{i-1} count_k + count_i = Counters[i]$ וסיימנו.

ניתוח סיבוכיות זמן

ניעזר בטבלה הבאה:

ההוראה	סיבוכיות זמן-ריצה
יצירת מערך המונים	$O(1)$
מעבר על כל איבר במערך וספירתו	$n * O(1) = O(n)$
מעבר על כל איבר במערך המונים והוספת האיבר שלפניו	$k - 1 = O(k)$

לסיכום, סיבוכיות זמן-הריצה של העיבוד המקדים היא $O(n) + O(k) = O(n + k)$.

כמו כן, ברור כי גישה למערך ופעולת החיסור הן בעלות סיבוכיות של $O(1)$, ולכן סיבוכיות זמן הריצה של השאילתא היא $O(1)$ כנדרש.

סעיף ב

בהינתן התפלגות מסוימת של n נקודות y_i, $i = 1..n$ בטווח $[0, 1]$, ניתן להכליל את הרעיון של מיון דלי ולמיין אותם ב**תוחלת זמן לינארית** על ידי n דליים, גם אם ההתפלגות שלהן אינה לגמרי אחידה, כפי שנדרש במיון דלי במקור.

ההתפלגות נתונה על ידי $k + 1$ נקודות x_i, $i = 0..k$, בטווח $[0, 1]$ הנתונות בסדר עולה ($x_0 \le x_1 \le ... \le x_k$), כך ש $x_0 = 0$, $x_k = 1$, ובכל קטע $[x_i, x_{i+1}]$ כאשר $i = 0..(k - 1)$, ההסתברות שאיבר בקלט יהיה שייך לקטע היא p_i. כמו כן, כמובן ש $\sum_{i=0}^{k-1} p_i = 1$. כמו כן נתון כי בכל קטע מסוג זה ההתפלגות היא אחידה.

זוהי הכללה של מיון דלי במקור: במיון דלי במקור, נדרש ש $k = 1$, ואז קיימות שתי נקודות והקטע ביניהן הוא $[0, 1]$. ההסתברות p_0 מקיימת $p_0 = \sum_{i=0}^{0} p_i = 1$ כנדרש, וההתפלגות בין שתי הנקודות היא אחידה כנדרש.

במיון דלי רגיל, אנחנו ממיינים את n הנקודות, הנתונות בקטע היחיד, בעזרת $p_0 \cdot n = n$ דליים.

בהתאם, במיון המוכלל שלנו, נמיין את מספר הנקודות הנתונות בכל קטע $[x_i, x_{i+1}]$ בעזרת $p_i \cdot n$ דליים. סך הכל, נמיין בעזרת $\sum_{i=0}^{k-1} p_i n = n \cdot \sum_{i=0}^{k-1} p_i = n \cdot 1 = n$ דליים.

כמו כן, בכל קטע נתון כי ההתפלגות היא אחידה. לכן, ההסתברות שאיבר בקלט יהיה שייך לדלי מסוים בקטע i היא $\frac{p_i}{p_i n} = \frac{1}{n}$, בדומה למיון דלי במקור!

נשאלת השאלה - איך נחלק את המערך לפי הקטעים בהתפלגות אליהם כל איבר שייך? הפתרון הוא לבצע סדרה של חלוקות דמויות partition, המחזירות את אינדקס האיבר האחרון הקטן מאיבר הציר.
האלגוריתם -

$Partition2(A,\ left,\ right,\ pivot)$
$\quad i \leftarrow left - 1 \triangleright Last\ index\ smaller\ than\ the\ pivot$
$\quad for\ j \leftarrow left..right$
$\quad\quad if\ A[j] < pivot\ then$
$\quad\quad\quad i \leftarrow i + 1$
$\quad\quad\quad swap(A, i, j)$
$\quad return\ i$

נכונות האלגוריתם נובעת ישירות מנכונות ה$Partition$. ההבדל היחיד הוא שאיברים השווים לאיבר הציר יהיו משמאל ולא מימין לחלוקה. בנוסף, נחזיר את אינדקס האיבר האחרון הקטן מהציר ולא את אינדקס האיבר הראשון הגדול או שווה לו.

כעת, משכל האיברים השייכים לאותו הקטע נמצאים יחדיו, איך נמיין אותם?
כלומר - איך ניתן למיין ערכים בקטע $[x_i, x_{i+1}]$, המוכל בקטע $[0, 1]$, באמצעות מיון דלי? לשם כך, ניצור ראשית $p_i \cdot n$ דליים. בכל קטע ישנה התפלגות אחידה, לכן עלינו להחליט לאיזה מבין הדליים האיבר שייך. יהא איבר y כלשהו השייך לקטע $[x_i, x_{i+1}]$. כל הדליים ה"קטנים" ממנו הם ה $\sum_{k=0}^{i-1} p_k n$

הדליים הקודמים. למספר זה יש להוסיף מספר בין 0 ל $p_i n$ על מנת להחליט על אינדקס הדלי הספציפי אליו האיבר שייך.

במיון דלי, בקטע $[0, 1]$, כאשר מספר הדליים "לפני" הוא 0, אנחנו קובעים מהו אינדקס זה על ידי כפל האיבר הנוכחי ב $p_0 n = n$.

באופן דומה, "נמתח" את הקטע $[x_i, x_{i+1}]$ לקטע $[0, 1]$, ונכפול כל איבר שם ב $p_i n$, ובכך ייקבע האינדקס שלו.

איך תתבצע המתיחה? ראשית, "נזיז" את x_i לנקודת האפס על ידי חיסור x_i מכל הנקודות בקטע.

לאחר מכן, את הקטע $[0, x_{i+1} - x_i]$ ניתן "למתוח" ל$[0, 1]$ על ידי חלוקה באורך הקטע - $(x_{i+1} - x_i)$.

המתיחה, המתוארת ע"י הפונקציה הלינארית $y = \frac{x - x_i}{x_{i+1} - x_i}$, שהשיפוע שלה $\frac{1}{x_{i+1} - x_i}$ הוא חיובי (המכנה חיובי כאורך קטע), היא מונוטונית עולה, ולכן הסדר היחסי בין האיברים אכן יישמר.

אכן יתקבל מספר בין $\sum_{k=0}^{i-1} p_k n$ ל $\sum_{k=0}^{i} p_k n = \sum_{k=0}^{i-1} p_k n + p_i n$ (כולל החסם העליון), כי לאחר המתיחה, נוסיף לסכום תוצאת כפל מספר בין 0 ו-1 ל$p_i n$, ובהוספת $\sum_{k=0}^{i-1} p_k n$ אכן נקיים את התנאי.

האלגוריתם המלא:

$Generalized\_Bucket\_Sort(A,\ Points,\ Probabilities)$

$\quad Buckets\ \leftarrow new\ DoubleList[A.length]$

$\quad occupied\_buckets \leftarrow 0 \triangleright representing\ \sum_{k=0}^{i-1} p_k n$

$\quad seq\_start \leftarrow 1$

$\quad \triangleright Make\ buckets$

$\quad for\ i \leftarrow 1..(Points.length\ -\ 1)$

$\quad\quad \triangleright Find\ end\ of\ current\ sequence$

$\quad\quad seq\_end \leftarrow Partition2(A,\ seq\_start,\ A.length,\ Points[i])$

$\quad\quad delta \leftarrow Points[i\ +\ 1]\ -\ Points[i]\ \triangleright Compute\ length\ of\ sequence$

$\quad\quad \triangleright Add\ sequence\ members\ to\ buckets$

$\quad\quad for\ j \leftarrow seq\_start..\ seq\_end$

$\quad\quad\quad bucket\ \leftarrow occupied\_buckets\ +\ \lceil \frac{(A[j]-Points[i])\cdot Probabilities(i)\cdot A.length}{delta} \rceil$

$\quad\quad\quad Buckets[bucket].addToList(A[j])$

$\quad\quad \triangleright Increase\ counters\ accordingly$

$\quad\quad occupied\_buckets \leftarrow occupied\_buckets\ +\ Probabilities(i)\ \cdot\ A.length$

$\quad\quad seq\_start\ =\ seq\_end\ +\ 1$

$\quad \triangleright Sort\ buckets\ and\ place\ in\ array$

$\quad for\ i \leftarrow 1..A.length$

$\quad\quad Insertion\_Sort(Buckets[i])$

$\quad concatenate\ the\ lists\ Buckets[0],\ Buckets[1],\ \dots\ , Buckets[n]\ together\ in\ order$

ניתוח נכונות

יהיו a, b שני איברים בקלט כך ש $a < b$.
טענה: בסוף ביצוע האלגוריתם, a יופיע לפני b במערך.
הוכחה: נפצל למקרים. ראשית, אילו קיים i כך שיש x_i מנקודות ההתפלגות הנתונות בין a ו-b

(במילים אחרות: אילו שני האיברים אינם שייכים לאותו קטע התפלגות) אז לאחר ה$partition$ הi, בוודאות a יופיע לפני b במערך.

לאחר מכן, בהתאם, a ייכנס לדלי בטווח שבין $\sum_{k=0}^{i-1} p_k n$ ל$(\sum_{k=0}^{i} p_k n - 1)$, ו b ייכנס לדלי

שהאינדקס שלו גדול מ$\sum_{k=0}^{i} p_k n$. בהתאם, בשלב שרשור הדליים למערך, a יופיע לפני b.

במקרה השני, a ו-b שייכים לאותו הקטע $[x_i, x_{i+1}]$.
נסמן $a = x_i + \alpha(x_{i+1} - x_i),\ b = x_i + \beta(x_{i+1} - x_i)$ כאשר מתקיים $0 \le \alpha < \beta < 1$.
לכן, אינדקס הדלי של a יהיה:

$$\sum_{k=0}^{i-1} p_k n + \frac{x_i + \alpha(x_{i+1} - x_i) - x_i}{x_{i+1} - x_i} \cdot p_i n = \sum_{k=0}^{i-1} p_k n + \frac{\alpha(x_{i+1} - x_i)}{x_{i+1} - x_i} \cdot p_i n = \sum_{k=0}^{i-1} p_k n + \alpha p_i n$$

וכן באופן דומה אינדקס הדלי של b יהיה $\sum_{k=0}^{i-1} p_k n + \beta p_i n$.

מאחר ו$\alpha < \beta$, נסיק כי $\sum_{k=0}^{i-1} p_k n + \alpha p_i n < \sum_{k=0}^{i-1} p_k n + \beta p_i n$ ובערך עליון\תחתון יתקיים אי-שוויון חלש. כלומר, a נמצא בדלי הקודם ל-b או באותו הדלי.

אילו הם נמצאים בדליים שונים, אז שוב בשלב שרשור הדליים למערך a יופיע לפני b. אחרת, שני האיברים יושוו ישירות במיון הדלי בו הם נמצאים, ו-a יופיע לפני b.

ניתוח סיבוכיות זמן ריצה

הפעולה	סיבוכיות זמן-ריצה
k פעמים, בצע:	$\Theta(1)$, k קבוע וידוע מראש.
$Partition$	$\Theta(n)$
הוספה לדליים של כל האיברים בסך הכל	$\Theta(n)$
מיון הדליים	במקרה הממוצע - $\Theta(n)$ כי לפי ההתפלגות יש מעט מאוד איברים בכל דלי בדומה למיון דלי במקור, במקרה הגרוע, תלוי בסוג המיון שהופעל, $\Theta(n^2)$ או $\Theta(n \lg n)$
מעבר על כל הדליים ופריסה מחדש לרשימה	$\Theta(n)$

לכן בסך הכל, ניתן למיין ב**תוחלת זמן לינארית** כנדרש!

שאלה 3

נשתמש בדו-התור, שנסמנו ב-Q, כדי לאחסן סדרה עולה של אינדקסים של איברים. לצורך פשטות, נקבע כי בהסברים על האלגוריתם, איברי התור ימוספרו משמאל לימין, כך ש $Left(Q) = i_1$.

נגדיר: לכל i ו-j כך ש $i \leq j$, דו-התור Q "<u>מכסה</u>" את תת-המערך $A[i..j]$ אם:
1. אינדקס האיבר המקסימלי בתת-המערך $A[i..j]$ הוא i_1.
2. לכל $m \geq 2$, אינדקס האיבר המקסימלי בתת-המערך $A[(i_{m-1} + 1)..j]$ הוא i_m.

הערה: אם ישנם שני ערכים מקסימליים, אז בתור יאחסן את האינדקס המאוחר יותר משניהם.
למשל, תור מסוים מכסה את המערך $[10, 8, 9, 6]$ אם הוא מכיל את הערכים הבאים: $[1, 3, 4]$, המייצגים את האינדקסים של הערכים במערך $10, 9, 6$.

רעיון האלגוריתם

בכל רגע נתון, בהינתן דו-תור המכסה את תת-המערך $A[i - k + 1, i]$, נוכל לענות על השאילתא $max(i - k + 1, i)$ ע"י גישה לאיבר i_1 בדו-התור ע"י $Left(Q)$, וגישה לאינדקס זה במערך, שכן $A[i_1]$ יהיה האיבר המקסימלי בתת-המערך ה "נוכחי".
1. נרוץ על k האיברים הראשונים במערך ונבנה דו-תור המכסה את תת-המערך $A[1..k]$
2. נדפיס את האיבר המקסימלי בתת-המערך $A[1..k]$
3. נרוץ על שאר איברי המערך, עד $i = n$, ונכניס ונוציא איברים מדו-התור על מנת שיכסה את תת-המערך $A[i - k + 1, i]$.
   בכל איטרציה נדפיס את האיבר המקסימלי בתת-המערך.

תיאור האלגוריתם

ראשית, נתאר את האלגוריתם לכיסוי האיבר ה $j + 1$, כאשר התור מכסה את תת-המערך $A[i, j]$

```
Enqueue_Element(A, Q, i)
    while not Q.isEmpty() and A[Right(Q)] ≤ A[i]
        PopRight(Q)
    PushRight(Q, i)
```

כעת, נתאר את האלגוריתם המלא:

```
Max_In_Subarrays(A, k)
    Q ← create empty bi − queue
    ▷ 1. Create query queue for the first k elements
    for i ← 1 to k
        Enqueue_Element(A, Q, i)
    ▷ 2. Print query for A[1..k]
    print(A[Left(Q)])
    ▷ 3. Go through the rest of the elements
    for i ← (k + 1) to A.length
        Enqueue_Element(A, Q, i) ▷ Queue now covers A[i − k to i]
        ▷ Extract the left element in the query queue if needed
        if Left(Q) = i − k
            PopLeft(Q)
        ▷ Queue now covers A[i − k + 1 to i]
        print(A[Left(Q)])
```

ניתוח נכונות

ראשית, נוכיח טענה חשובה על ההגדרה שהגדרנו.

טענה: בכל תור מוגדר היטב בעל $p \geq 1$ איברים המכסה תת-מערך כלשהו $A[i..j]$, מתקיים $A[i_1] > ... > A[i_p]$.

הוכחה: נוכיח כי לכל m המקיים $1 \leq m \leq p - 1$, מתקיים $i_m > i_{m+1}$.

לפי ההגדרה, i_m הוא אינדקס האיבר המקסימלי בתת מערך החלקי ל $A[i..j]$.

לכן, ברור ש $i \leq i_m \leq j$.

כמו כן, $i_m + 1 \leq i_{m+1} \leq j$ לפי הגדרה. לכן, ברור שהאיבר ה i_{m+1} במערך מוכל בתת-המערך ש-i_m הוא אינדקס האיבר המקסימלי בו, אבל $i_m + 1 \leq i_{m+1}$ ולכן $i_{m+1} \neq i_m$.

כלומר, $A[i_{m+1}]$ הוא איבר בתת-מערך ש $A[i_m]$ הוא האיבר המקסימלי בו, והאינדקסים שלהם שונים זה מזה, לכן $A[i_{m+1}] < A[i_m]$. (אילו היה מתקיים שוויון, אז לפי ההגדרה $i_m = i_{m+1}$ וזאת בסתירה מוחלטת לבחירת i_{m+1}).

כעת, ננתח את נכונותה של השגרה הפנימית $Enqueue\_Element(A, Q, i)$.

טענה: בהינתן תור ריק, הפעלת השגרה עם אינדקס כלשהו i תגרום לכך שהתור יכסה את תת-המערך $A[i..i]$.

הוכחה: אילו התור ריק, אז תנאי הלולאה אינו מתקיים.

האיבר i ייכנס לתור ויהיה האיבר היחיד בו. לכן $i_1 = i$ וברור ש-i הוא אינדקס האיבר המקסימלי בתת-מערך כלשהו המכיל רק את עצמו.

טענה: בהינתן תור המכסה תת-מערך כלשהו $A[i..j]$, הפעלת השגרה כאשר $i = j + 1$ תגרום לכך שהתור יכסה את תת-המערך $A[i..(j + 1)]$.

הוכחה: יהא תור המכסה את תת-המערך $A[i..j]$. בתור יש לפחות איבר אחד, שכן תת-המערך אותו התור מכסה אינו ריק וחייב להיות בו איבר מקסימלי. כמו כן, בתור יש $p \geq 1$ איברים.

אילו $A[j + 1] < A[i_p]$, אז תנאי הלולאה מעולם לא יתקיים, והאלגוריתם ידחוף לתור מימין את האינדקס $j + 1$.

מאחר ולפי טענת העזר $A[i_1] > ... > A[i_p]$, אז לפי טרנזיטיביות לכל m מתקיים $A[i_m] > A[j + 1]$. לפי ההגדרה $A[i_m]$ איבר מקסימלי בתת-המערך המתחיל באינדקס i_{m-1} (או i) ונגמר באינדקס j, ומאי השוויון $A[i_m] > A[j + 1]$ נסיק כי $A[i_m]$ גם איבר מקסימלי בתת-המערך המתחיל באינדקס i_{m-1} ונגמר באינדקס $j + 1$.

מאחר וטענה זו מתקיימת לכל m, סיימנו.

אחרת, יוצאו החוצה איברים עד שהתנאי יתקיים עד m כלשהו.

כעת, מאחר ו $A[j + 1] \geq A[i_{m+1}]$, ואיבר זה הוא האיבר המקסימלי בתת-המערך $A[(i_m + 1)..j]$, אז $A[j + 1]$ האיבר המקסימלי בתת-המערך $A[(i_m + 1)..(j + 1)]$ כנדרש.

לבסוף, ננתח את הנכונות של השגרה המרכזית.
שמורת לולאה: לאחר האיטרציה ה-i של הלולאה הראשונה, התור מכסה את תת-המערך $A[1..i]$.
נוכיח באינדוקציה.
<u>בסיס האינדוקציה:</u> לאחר האיטרציה הראשונה, לפי הטענות הקודמות, האלגוריתם קרא לשגרה $Enqueue\_Element$ עם תור ריק ו$i = 1$ ולכן התור מכסה את תת-המערך $A[1..1]$ כנדרש.
<u>צעד האינדוקציה:</u> נניח כי באיטרציה ה$i - 1$ התור כיסה את תת-המערך $A[1..(i - 1)]$.
כעת, האלגוריתם קרא לשגרה $Enqueue\_Element$ עבור $i = (i - 1) + 1$ ולכן לפי הטענות הקודמות, התור יכסה כעת את תת-המערך $A[1..i]$ כנדרש.
<u>סיום:</u> בסיום הלולאה, האלגוריתם יכסה את תת-המערך $A[1..k]$.

שמורת לולאה: לאחר האיטרציה הi של הלולאה השנייה, התור מכסה את תת-המערך $A[i - k + 1, i]$.
נוכיח באינדוקציה.
<u>בסיס האינדוקציה:</u> בכניסה ללולאה, לפי נכונות השמורה הקודמת, התור מכסה את תת-המערך $A[1..k]$. לאחר מכן, מנכונות השגרה $Enqueue\_Element$ עבור $i = k + 1$, התור יכסה את תת-המערך $A[1..(k + 1)]$.
כעת, אילו $i_1 = 1$, אז נוציא מהתור (משמאל) את i_1. לפי ההגדרה, i_2 (לפני ההוצאה) הוא אינדקס האיבר המקסימלי בתת-המערך $A[(i_1 + 1)..(k + 1)]$, כלומר $A[2..(k + 1)]$.
לאחר ההוצאה איבר זה יהפך להיות האיבר הראשון בתור, ונקבל שהתור יקיים את התנאי הראשון בהגדרה. ברור שהתנאי השני מתקבל, כי תתי-המערכים מימין לאינדקס i_2 לא השתנו בכלל כתוצאה מהוצאת האיבר הראשון. לכן, התור מכסה את תת-המערך $A[2..(k + 1)]$ כנדרש.
אחרת, אילו $i_1 \neq 1$, האלגוריתם לא יבצע דבר. ברור כי במקרה זה i_1 מקיים $2 \leq i_1 \leq k + 1$ ולכן התור מכסה את תת-המערך $A[2..(k + 1)]$ כנדרש גם במקרה זה.
<u>צעד האינדוקציה:</u> אנלוגי לחלוטין לבסיס האינדוקציה. בבסיס האינדוקציה הנחנו כי התור מכסה את תת-המערך $A[1..k]$ והוכחנו עבור $A[2..(k + 1)]$, וכעת מניחים עבור $A[(i - k)..(i - 1)]$ כלשהו המקיים $k + 2 \leq i \leq n$, ומוכיחים עבור $A[(i - k + 1)..i]$.

לסיכום, ננתח את נכונות האלגוריתם.
1. לפי שמורת הלולאה הראשונה, בשלב הראשון אכן נבנה תור המכסה את תת-המערך $A[1..k]$.
2. מנכונות השלב הראשון, בשלב השני יודפס $A[i_1]$, שהוא האיבר המקסימלי בתת-המערך $A[1..k]$ כנדרש.
3. לאחר מכן, לפי שמורת הלולאה השנייה, בכל איטרציה דואגים לכך שהתור מכסה תת-מערך בגודל k מסוים, ואז מדפיסים את $A[i_1]$ - האיבר המקסימלי בתת-מערך זה.

ניתוח סיבוכיות זמן ריצה

ראשית, ננתח באופן כולל את סיבוכיות זמן-הריצה של כל הקריאות ל $Enqueue\_Element$.
בסך הכל, מכניסים לתור האינדקסים בדיוק n אינדקסים, מתוכם k בלולאה הראשונה ו-$(n-k)$ בלולאה השנייה. כל הכנסה כזאת, מחירה $\Theta(1)$.
כמו כן, מוציאים לכל היותר n איברים מהתור. לא ניתן להוציא אף איבר שאינו הוכנס, ולכן בסך הכל הוצאו לכל היותר n איברים. כל הקריאות לשגרה זו מחירן ביחד $n \cdot \Theta(1) + O(n) \cdot \Theta(1) = \Theta(n)$

ננתח את סיבוכיות זמן-הריצה של השגרה המרכזית

הצעד	סיבוכיות זמן-ריצה
בניית תור האינדקסים מ-k האיברים הראשונים	*נכלל ב $\Theta(n)$ סכום הצעדים ש $Enqueue\_Element$ מבצעת*
הדפסה ראשונה	$\Theta(1)$
מעבר על שאר איברי המערך וביצוע:	$\Theta(n-k)$
הכנסת אינדקס לתור האינדקסים	*נכלל ב $\Theta(n)$ סכום הצעדים ש $Enqueue\_Element$ מבצעת*
בדיקה והוצאה של איבר מהתור אם יש צורך	$\Theta(1)$
הדפסה	$\Theta(1)$

בסך הכל - $\Theta(n)$ סיבוכיות זמן-ריצה.

שאלה 4

ראשית, לפי ההנחה, לכל מספר טבעי ניתן לחשב את סכום ספרותיו ב $\Theta(1)$. לכן נגדיר פונקציה $sumOfDigits(n)$ המחזירה את סכום הספרות של המספר הטבעי הניתן לה כקלט בזמן קבוע.

בהינתן קלט בין n מספרים טבעיים, יש מספר סופי, הקטן או שווה ל-n, של סכומי ספרות. כל סכום ספרות נמצא בין 1, סכום הספרות הקטן ביותר האפשרי למספר טבעי, וסכום הספרות הגבוה ביותר של מספר בקלט, אותו ניתן לחשב בקלות ב $\Theta(n)$:
בהינתן קלט של n מספרים טבעיים A:

$maxSumOfDigits(A)$
$max \leftarrow 0$
$for\ i \leftarrow 1\ to\ A.length$
$curr \leftarrow sumOfDigits(A[i])$
$if\ curr\ >\ max\ then$
$max \leftarrow curr$
$return\ max$

ניתן להוכיח בעזרת שמורת לולאה כי בסוף האיטרציה הi, במשתנה max יש את סכום הספרות הגבוה ביותר מבין סכומי הספרות של המספרים הטבעיים $A[1],...,A[i]$. לכן, לאחר ביצוע הלולאה, במשתנה max יישמר סכום הספרות הגבוה ביותר בקלט, וזהו פלט האלגוריתם.
כמו כן, מההנחה שסיבוכיות זמן הריצה של $sumOfDigits(n)$ היא $\Theta(1)$, בגוף הלולאה מתבצעות הוראות בסיבוכיות זמן קבועה בלבד. לכן, בסך הכל, הלולאה תתבצע בזמן-ריצה לינארי. נסמן ב-m את סכום ספרות זה.

כעת, נרצה לבנות טבלת גיבוב עבור המספרים בקלט, כך שמפתחות הטבלה יהיו מהקבוצה $U = \{1,...,m\}$, הקטנה מהרבה מקבוצת כל סכומי הספרות האפשריים. פונקציית הגיבוב תהיה $sumOfDigits$, והיא אכן מחזירה ב$\Theta(1)$ מפתח המתאים למספר הטבעי הניתן לה, בטווח שבין 1 ל-m בכל תא בטבלה, תישמר רשימה מקושרת של האיברים שתמונת הגיבוב שלהם, כלומר סכום ספרותיהם, הוא כנדרש. נניח כי לרשימה קיימת תכונה בשם $size$ העוזרת לדעת את כמות האיברים ברשימה בכל עת, ונשתמש בתכונה זו על מנת לקבוע איזה סכום ספרות מופיע מספר מקסימלי של פעמים בין איברי הקלט.

<u>הערה:</u> אם נרצה לשמור על הסדר היחסי בין האיברים בקלט, אז נניח שהרשימה שומרת גם מצביע לסוף כך שפעולת הוספה לסוף הרשימה תיקח $\Theta(1)$. כך או כך, ננניח כי קיימת פעולת $List.append()$ המוסיפה איבר לרשימה (בהתחלה או בסוף, תלוי בקיום המצביע לזנב) ב $\Theta(1)$.

רעיון האלגוריתם:
1. נמצא את סכום הספרות המקסימלי
2. ניצור טבלת גיבוב שעולם המפתחות שלה הוא $\{1,...,m\}$ ופונקציית הגיבוב שלה היא $sumOfDigits\colon\ \{A[i]\ |\ 1\ \le\ i\ \le\ n\} \rightarrow \{1,...,m\}$. (ניתן להפחית 1 על מנת להשתמש בתאים 0 עד $m\ -\ 1$ במערך, אבל לשם פשטות נוותר על ההוספות וההחסרות של 1 באלגוריתם זה ונניח שמערך הטבלה T מתחיל באינדקס 1). נכנסי לטבלה את כל איברי הקלט, ותוך כך נמצא את התא בטבלה בעל מספר האיברים הרב ביותר.
3. נדפיס את איברי הטבלה בתא זה

נשים לב שלא הוגדר לנו מה לעשות כאשר יש יותר משני סכומי ספרות הנפוצים באותה המידה. במקרה כזה, נחזיר אחד באופן שרירותי.

האלגוריתם, עבור מערך של מספרים טבעיים A:

$mostFrequentSumOfDigits(A)$

$m \leftarrow maxSumOfDigits(A)$

$table\ = new\ array[m]\ of\ LinkedList$

$\triangleright\ Iterate\ through\ the\ array\ and\ populate\ the\ table,\ keep\ a\ pointer\ to\ the\ max\ index$

$maxIndex \leftarrow sumOfDigits(A[1])$

$for\ i \leftarrow 1\ to\ A.length$

$curr \leftarrow sumOfDigits(A[i])$

$table[curr].append(A[i])$

$if\ table[curr].size\ >\ table[maxIndex].size$

$maxIndex \leftarrow curr$

$\triangleright\ print\ elements\ of\ table[maxIndex]$

$p \leftarrow table[max].head$

$while\ p\ \neq\ null$

$print(p.key)$

$p \leftarrow p.next$

ניתוח נכונות:

1. כבר ניתחנו את נכונות השלב הראשון באלגוריתם
2. בשלב השני, נכניס איברים לטבלה ונשמור את אינדקס התא בעל המספר הרב ביותר של איברים שנכנסו עד כה.
בעזרת הטענה שתוכח בהמשך, "לאחר כל איטרציה של הלולאה המרכזית, תכונת ה$size$ של המשתנה בתא $table[maxIndex]$ תהיה גדולה מתכונות ה$size$ של כל התאים האחרים", נסיק כי לאחר הלולאה השנייה תכונת ה $size$ של $table[maxIndex]$ היא הגדולה ביותר מתכונות ה $size$ של כל תאי המערך. כלומר, בתא $table[maxIndex]$ יש הכי הרבה איברים מבין כל איברי המערך.

לכן, מהגדרת פונקציית הגיבוב, סכום הספרות הנפוץ ביותר בקלט הוא $maxIndex$, והאיברים שסכום ספרויתיהם הוא $maxIndex$ שמורים ברשימה ב$table[maxIndex]$.

טענה: לאחר כל איטרציה של הלולאה המרכזית, תכונת ה$size$ של המשתנה בתא $table[maxIndex]$ תהיה גדולה מתכונות ה$size$ של כל התאים האחרים.

הוכחה: נוכיח את הטענה באינדוקציה על מספר האיטרציה.

<u>בסיס</u>: לפני האיטרציה הראשונה מתקיים $maxIndex = sumOfDigits(A[1])$, וכן כל הרשימות המקושרות בטבלה ריקים (כלומר מתקיים $size = 0$)

עבור $i = 1$, יוכנס האיבר $A[i]$ לתא המתאים בטבלה. כעת, בתא זה, שהאינדקס שלו הוא $maxIndex = sumOfDigits(A[1])$, יש איבר אחד בדיוק, מספר גבוה יותר משאר הרשימות.

<u>צעד האינדוקציה:</u> נניח כי הטענה נכונה עבור i כלשהו. באיטרציה הבאה נוסיף איבר כלשהו לתא ה$sumOfDigits(A[i + 1])$ במערך.

כעת, אילו $table[sumOfDigits(A[i + 1])].size > table[maxIndex].size$, אז לפי טרנזיטיביות בתא אליו הכנסנו איבר יש יותר איברים מכל תא אחר בטבלה

לכן תבוצע ההוראה $maxIndex \leftarrow sumOfDigits(A[i + 1])$ וסיימנו.

אחרת, ב$table[maxIndex]$ יש עדיין את מספר האיברים המקסימלי. לא יבוצע דבר וסיימנו.

ניתוח סיבוכיות זמן-ריצה

הפעולה	סיבוכיות זמן-ריצה
מציאת סכום הספרות המקסימלי	$\Theta(n)$ כמפורט קודם.
מעבר על כל איברי הקלט, הכנסה לטבלת הגיבוב, וביצוע פעולת השוואה.	$n \cdot \Theta(1) = \Theta(n)$, בהתבסס על העובדה כי הכנסה לרשימה מקושרת, פונקציית הגיבוב ופעולות ההשוואה הן בעלות סיבוכיות קבועה.
הדפסת איברי הרשימה	$\Theta(1 + \alpha)$ במקרה הממוצע, ובכל מקרה לכל היותר n איברים ולכן במקרה הגרוע $\Theta(n)$.